Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently amended) A system for analyzing thread deadlocks in a Java virtual machine, comprising:

a thread analyzer tool, wherein the tool analyzes a thread dump to automatically identify thread deadlocks[[;]] , wherein the tool identifies threads that are in a self wait condition and threads that are in a circular wait condition; and

a user interface that allows a user to specify criteria, wherein the tool excludes threads that do not meet the criteria from being identified on the user interface as deadlocked threads even though the threads were identified as being deadlocked threads pursuant to the thread dump analysis.

2. (Cancelled)

3. (Original) The system of claim 1, wherein the tool analyzes a thread dump file in offline mode.

4. (Original) The system of claim 1, wherein the tool obtains a thread dump from a running information processing system.

5. (Currently amended) A method of analyzing thread deadlocks in a Java virtual machine, comprising the steps of:

obtaining a thread dump of a Java virtual machine;

analyzing the thread dump to automatically identify threads in a deadlock condition[[;]] , wherein threads in a circular wait condition and threads in a self wait condition are identified; and

receiving, by a user interface, user specified criteria from a user, wherein identified deadlock threads that do not meet the criteria are filtered such that they are not presented to the user by the user interface even though the threads that do not meet the criteria were identified as being deadlocked threads pursuant to the thread dump analysis.

6. (Cancelled)

7. (Currently amended) The method of claim [[6]] 5, wherein filtered threads are excluded from identification as threads in a self wait condition.

8. (Original) The method of claim 5, wherein the tool analyzes a thread dump file in offline mode.

9. (Currently amended) The method of claim 5, wherein a matrix is populated [[with]] to identify threads owning resources and threads waiting on resources, and wherein the matrix is used to identify threads in a circular wait condition.

10. (Currently amended) The method of claim 5, wherein the step of analyzing the thread dump to automatically identify threads in a deadlock condition includes the steps of:

(a) identifying, in the thread dump, a locked object that is already in use by a thread;

[[(a)]] (b) identifying threads that own resources the thread that has locked the locked object;

[[(b)]] (c) identifying each waiting thread ~~threads~~ that [[are]] is waiting on ~~resources~~ the locked object; and

(d) comparing the results from steps ~~(a) and~~ (b) and (c), and then repeating steps (a)-(c) for each locked object in the thread dump, to identify the threads in [[a]] the circular wait condition and the self wait condition.

11. (Original) A method of analyzing thread deadlocks in a Java virtual machine (JVM), comprising the steps of:

obtaining a thread dump file;

identifying waiting threads;

identifying locking threads; and

comparing waiting threads and locking threads to identify threads in a self wait condition.

12. (Original) The method of claim 11, further comprising the step of comparing waiting threads and locking threads to identify threads in a circular wait condition.

13. (Original) The method of claim 11, wherein the step of obtaining a thread dump file comprises obtaining a thread dump from a live JVM.

14. (Original) The method of claim 11, wherein the step of obtaining a thread dump file comprises opening an existing thread dump file.

15. (Original) The method of claim 14, wherein the existing thread dump file is analyzed in offline mode.

16. (Original) The method of claim 11, wherein a user interface allows a user to choose rules, and wherein the rules are used to exclude threads from being identified as in a deadlock condition.

17. (Original) A system for analyzing thread deadlocks in a Java virtual machine (JVM), comprising the steps of:

means for obtaining a thread dump file;

means for identifying waiting threads;

means for identifying locking threads; and

means for comparing waiting threads and locking threads to identify threads in a self wait condition.

18. (Original) The system of claim 17, further comprising means for comparing waiting threads and locking threads to identify threads in a circular wait condition.

19. (Original) The system of claim 17, wherein thread deadlocks are analyzed in offline mode.

20. (Currently amended) A computer program product data structure encoded in a computer readable medium for analyzing thread deadlocks in a Java virtual machine, comprising:

first instructions for obtaining a thread dump file;

second instructions for identifying waiting threads;

third instructions for identifying locking threads; and

fourth instructions for comparing waiting threads and locking threads to identify threads in a self wait condition.

21. (Original) The computer program product of claim 20, further comprising fifth instructions for comparing waiting threads and locking threads to identify threads in a circular wait condition.

22. (Original) The computer program product of claim 20, wherein thread deadlocks are analyzed in offline mode.